DATA STRUCTURES LABORATORY WITH APPLICATIONS

Course code: 19CS3DLDSL Credits: 02 L: P: T: S: 0:2:2:0 CIE Marks: 50 Exam Hours: 03 SEE Marks: 50 Total Hours: 50

Course objectives:

1. Introduce the concept of data structures through ADT including List, Stack, Queues
2. To design and implement various data structure algorithms.
3. To introduce various techniques for representation of the data in the real world
4. To develop application using data structure algorithms

Course Outcomes: At the end of the course, students will be able to:

CO1	Apply appropriate data structures as applied to specified problem definition
CO2	Perform operations like searching, insertion, and deletion, traversing mechanism etc. on various data structures.
CO3	Implement Linear and Non-Linear data structures.
CO4	Implement various dynamic node insert and delete operations on Linked Lists
CO5	Demonstrate the data structure using Non-Linear data structure.
CO6	Demonstrate of tree operation using dynamic node addition and deletion.

Expt No	**Content of the Experiment**	**Hours**
1a.	Program 1: Write a program in C to simulate the working of a stack of integers using an array. Provide the following operations. a. Insert b. Delete c. Display	02
1b.	Application: In a calculator application, given an expression with parenthesis it is required to flag any parenthesis mismatch. Develop a program for this calculator application using appropriate data structure.	
2a.	Program 2: Design, develop and execute a program in C to simulate the working of a queue of integers using an array. Provide the following operations. a. Insert b. Delete c. Display	02

2b.	Application: In a system, resources are shared among multiple consumers for optimal performance. Considering jobs submitted to the printer have to be printed in the order of arrival. In print spooling, documents are loaded into a buffer and then the printer pulls them off the buffer at its own rate. Spooling also lets you place a number of print jobs on a queue instead of waiting for each one to finish before specifying the next one. Develop a program for such scheduling using appropriate data structure.	

3a.	Program 3: Design, develop and execute a program in C to perform the following operation on Singly linked list a. Insert the node at a specific location b. Delete a node at the begining of the list c. Display the list	02
3b.	Application: Scientific applications involve many polynomials with varied degree. A polynomial is an expression consisting of variables and coefficients. Choose an appropriate data structure to represent the polynomial and perform addition of two polynomials..	
4a.	Program 4: Design, develop and execute a program in C to implement a doubly linked list where each node consists of integers. The program should support the following operations: a. Create a doubly linked list by adding each node at the front b. Delete the node of a given data If it is found, otherwise display appropriate message c. Display the contents of the list.	02
4b.	Application: Colleges maintain the student’s records. As students enroll and graduate, the records have to be updated. It is necessary to access the student’s data whenever required by college. Using suitable data structures develop a program for above scenario.	
5a.	Program 5: Design, develop and execute a program in C to create a Binary tree using arrays and display the tree.	02
5b.	Application: Compilers use expression trees to represent mathematical expressions where the leaf nodes represent the operands and the internal nodes represent the operators. Develop a program to evaluate such an expression tree with non-negative integers as operands and the arithmetic operators ‘+’, ‘-‘, ‘*’and ‘/’.	
6a.	Program 6: Design, develop and execute a program in C to create a Binary search tree and perform preorder traversals.	02

6b.	Application : In a payroll management system, it is required to store the employee data (Employee ID, Employee name, Login Time) as one logs in to the system. At the end of the day, it is required to generate a report of all the employee who logged in that day in ascending order of the Employee ID. Develop a program to generate this report.	

Text Book:

1. Fundamentals of Data Structures, Sartaj Sahni, University Press

Reference Books:

1. Yedidyah, Augenstein, Tannenbaum: Data Structures Using C and C++, 2nd Edition, Pearson Education, 2003.
2. Richard F. Gilberg and Behrouz A. Forouzan: Data Structures A Pseudocode Approach with C, Cengage Learning, 2005.
3. A.M Padma Reddy,” Approach of Data Structures”, Person Publication, 5th Edition, 2015
4. ReemaTheraja “ Data Structure using C. 1st Edition , 2014

1a. Program 1: Write a program in C to simulate the working of a stack of integers using an array. Provide the following operations. a. Insert b. Delete c. Display

```
#include<stdio.h>
#define STACK_SIZE 3
int s[STACK_SIZE];
int top=-1;

void push()
{
        int n;
        if(top==STACK_SIZE-1)
                printf("\nStack overflow\n");
        else
        {
                printf("\nEnter the data to be pushed\n");
                scanf("%d",&n);
                s[++top]=n;
        }
}
void pop()
{
        if(top==-1)
                printf("\nStack empty\n");
        else
                printf("\n%d is popped\n", s[top--]);
}

void display()
{
        int i;
        if(top==-1)
                printf("\nStack empty\n");
        else
        {
                printf("\nStack elements are\n");
                for(i=top;i>=0;i--)
                        printf("%d\n",s[i]);
        }
}

int main()
{
        int ch;
        for(;;)
        {
                printf("\n1.PUSH\t2.POP\t3.DISPLAY\t4.EXIT\n");
                printf("\nEnter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: push();
                                        break;

                        case 2: pop();
                                        break;

                        case 3: display();
                                        break;

                        case 4: return 0;

                        default: printf("\nInvalid choice\n");
                }
        }
}
```

OUTPUT

```
1.PUSH 2.POP 3.DISPLAY 4.EXIT
Enter your choice
1
Enter the data to be pushed
10

1.PUSH 2.POP 3.DISPLAY 4.EXIT
Enter your choice
1
```

```
Enter the data to be pushed
20

1.PUSH 2.POP 3.DISPLAY 4.EXIT
Enter your choice
1
Enter the data to be pushed
30

1.PUSH 2.POP 3.DISPLAY 4.EXIT
Enter your choice
3
Stack elements are
30
20
10

1.PUSH 2.POP 3.DISPLAY 4.EXIT
Enter your choice
2
30 is popped
```

1b. Application: In a calculator application, given an expression with parenthesis it is required to flag any parenthesis mismatch. Develop a program for this calculator application using appropriate data structure.

```
#include <stdio.h>
#define sz 20
#include <string.h>
void push (char s[sz],int *top,char ch)
{
        *top = *top+1;
        s[*top] = ch;
}
void pop (int *top)
{
        *top = *top-1;
}
int main()
{
        char in[sz],ch,s[sz];
        int i,top=-1;
        printf ("Enter the Expression\n");
        scanf ("%s", in);
        push (s,&top,'#');
        for (i=0;i<strlen(in);i++)
        {
                ch = in[i];
                if (ch == '(')
                push (s,&top,ch);
                if (ch == ')')
                {
                        if (s[top] != '#')
                                pop (&top);
                        else
                        {
                                printf( "Closing Parentheses are not balanced\n");
                                return (0); }
                }
        }
        if (s[top] == '#')
                printf ("Parentheses are balanced\n");
        else
                printf ("Opening Parentheses are not balanced\n");
}
```

Output

```
1. Enter the Expression
(((a+b)-c)/d)
Parentheses are balanced
2. Enter the Expression
((((a+b)-c)/d
Opening Parentheses are not balanced
3. Enter the Expression
(a+b)-c)/d)
Closing Parentheses ae not balanced
```

2a. Program 2: Design, develop and execute a program in C to simulate the working of a queue of integers using an array. Provide the following operations. a. Insert b. Delete c. Display

```
#include<stdio.h>
#define maxsize 3
int q[maxsize], front=0,rear=-1;

void insert()
{
        int n;
        if(rear==maxsize-1)
                printf("\nQueue full\n");
        else
        {
                printf("\nEnter the data to be added\n");
                scanf("%d", &n);
                q[++rear]=n;
        }
}

void delete()
{
        if(front>rear)
                printf("\nQueue is empty\n");
        else
        {
                printf("\n%d is deleted\n",q[front++]);
```

```
		if(front>rear && rear==maxsize-1)
		{
			printf("\nReinit\n");
			front=0; rear=-1;
		}
	}
}

void display()
{
	int i;
	if(front>rear)
		printf("\nQueue is empty\n");
	else
	{
		printf("\nQueue status is\n");
		for(i=front;i<=rear;i++)
			printf("%d\t",q[i]);
	}
}

int main()
{
	int ch;
	while(1)
	{
		printf("1.Insert\n2.Delete\n3.Display\n4.Exit\n");
		puts("\nEnter your choice\n");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1: insert(); break;
			case 2:delete(); break;
			case 3:display(); break;
			case 4: return 0;
			default :printf("\nInvalid choice\n");
}
	}
}
```

OUTPUT

```
1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
1
Enter the data to be added
10

1.Insert
```

```
2.Delete
3.Display
4.Exit
Enter your choice
1
Enter the data to be added
20

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
3
Queue status is
10 20

1.Insert
2.Delete
3.Display
4.Exit
Enter your choice
2
10 is deleted
```

2b. Application: In a system, resources are shared among multiple consumers for optimal performance. Considering jobs submitted to the printer have to be printed in the order of arrival. In print spooling, documents are loaded into a buffer and then the printer pulls them off the buffer at its own rate. Spooling also lets you place a number of print jobs on a queue instead of waiting for each one to finish before specifying the next one. Develop a program for such scheduling using appropriate data structure.

```
#include <stdio.h>
#define sz 100
#include <stdlib.h>
void insert (int cq[sz],int *rear,int item,int *count)
{
        *rear = (*rear+1)%sz;
        cq[*rear] = item;
        *count = *count+1;
}
void deletion (int cq[sz],int *front,int *count)
{
        if (*count == 0)
                printf ("No Job in Print Spool\n");
        else
        {
                printf ("%d is Exits Print Spool\n",cq[*front]);
                *front = (*front+1)%sz;
                *count = *count-1;
        }
}
```

```
void display (int cq[sz],int front,int count)
{
    int i,j;
    if (count == 0)
        printf ("Print Spool is Empty\n");
    else
    {
        printf ("Content of Print Spool\n");
        j = front;
        for (i=1;i<=count;i++)
        {
            printf ("%d is printing\n",cq[j]);
            j = (j+1)%sz;
        }
    }
}
int main ()
{
    int cq[sz],rear=-1,count=0,front=0,item,ch;
    for (;;)
    {
        printf ("1:INSERT 2:DELETE 3:DISPLAY\n");
        scanf ("%d",&ch);
        switch (ch)
        {
            case 1: printf ("Enter the Print job in Spool\n");
                scanf ("%d",&item);
                insert (cq,&rear,item,&count);
                break;
            case 2: deletion( cq,&front,&count);
                break;
            case 3: display (cq,front,count);
                break;
            default: exit(0);
        }
    }
    return(0);
}
```

OUTPUT

```
1:INSERT 2:DELETE 3:DISPLAY
2
No Job in Print Spool
1:INSERT 2:DELETE 3:DISPLAY
3
Print Spool is Empty
1:INSERT 2:DELETE 3:DISPLAY
1
Enter the Print job in Spool
```

```
7
1:INSERT 2:DELETE 3:DISPLAY
1
Enter the Print job in Spool
3
1:INSERT 2:DELETE 3:DISPLAY
1
Enter the Print job in Spool
1
1:INSERT 2:DELETE 3:DISPLAY
3
Content of Print Spool
7 is printing
3 is printing
1 is printing
1:INSERT 2:DELETE 3:DISPLAY
2
7 is Exits Print Spool
1:INSERT 2:DELETE 3:DISPLAY
1
Enter the Print job in Spool
8
1:INSERT 2:DELETE 3:DISPLAY
1
Enter the Print job in Spool
5
1:INSERT 2:DELETE 3:DISPLAY
3
Content of Print Spool
3 is printing
1 is printing
8 is printing
5 is printing
1:INSERT 2:DELETE 3:DISPLAY
2
3 is Exits Print Spool
1:INSERT 2:DELETE 3:DISPLAY
2
1 is Exits Print Spool
1:INSERT 2:DELETE 3:DISPLAY
2
8 is Exits Print Spool
1:INSERT 2:DELETE 3:DISPLAY
2
5 is Exits Print Spool
1:INSERT 2:DELETE 3:DISPLAY
2
No Job in Print Spool
```

3a. Program 3: Design, develop and execute a program in C to perform the following operation on Singly linked list

a. Insert the node at a specific location

b. Delete a node at the beginning of the list
c. Display the list

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int info;
        struct node *link;
};
typedef struct node *NODE;
NODE insertLoc(NODE first)
{
        int loc,count;
        NODE temp,cur;

        printf("\nEnter the location\n");
        scanf("%d",&loc);
        temp = (NODE)malloc(sizeof(struct node));
        printf("\nEnter the data\n");
        scanf("%d",&temp->info);
        temp->link=NULL;

        if(first==NULL)
        {
                if(loc==1)
                        first = temp;
                else
                        printf("Invalid location\n");
        }
        else if(loc==1)
        {
                temp->link=first;
                first=temp;
        }
        else
        {
                cur=first;
                count=1;
                while(cur!=NULL)
                {
                        if(count==loc-1)
                        {
                                temp->link=cur->link;
                                cur->link=temp;
                                break;
                        }
                        cur=cur->link;
                        count++;
                }
                if(cur==NULL)
```

```
            printf("Invalid location\n");
    }
    return first;
}
NODE delete (NODE first)
{
    NODE temp;
    if (first == NULL)
    {
        printf ("List Empty\n");
        return first;
    }
    temp = first;
    first = first->link;
    printf ("%d is deleted\n",temp->info);
    free (temp);
    return first;
}
void display (NODE first)
{
    NODE temp;
    if (first == NULL)
        printf ("List is Empty\n");
    else
    {
        printf ("Content of List\n");
        temp = first;
        while (temp != NULL)
        {
            printf ("%d\t",temp->info);
            temp = temp->link;
        }
        printf ("\n");
    }
}
int main ()
{
    int ch;
    NODE first = NULL;
    for (;;)
    {
        printf ("1:INSERT 2:DELETE 3:DISPLAY 4.EXIT\n");
        scanf ("%d",&ch);
        switch (ch)
        {
            case 1: first = insertLoc (first);
                    break;
            case 2: first = delete (first);
                    break;
            case 3: display (first);
                    break;
```

```
                              default: exit(0);
                    }
          }
}
```

OUTPUT

```
1:INSERT 2:DELETE 3:DISPLAY
4.EXIT 1

Enter the location
1

Enter the data
10
1:INSERT 2:DELETE 3:DISPLAY
4.EXIT 1

Enter the location
2

Enter the data
20
1:INSERT 2:DELETE 3:DISPLAY
4.EXIT 1

Enter the location
3

Enter the data
30
1:INSERT 2:DELETE 3:DISPLAY 4.EXIT
3
Content of List
10 20 30
1:INSERT 2:DELETE 3:DISPLAY
4.EXIT 1

Enter the location
5

Enter the data
50
Invalid location
1:INSERT 2:DELETE 3:DISPLAY
4.EXIT 3
Content of List
10 20 30
1:INSERT 2:DELETE 3:DISPLAY
4.EXIT 2
10 is deleted
1:INSERT 2:DELETE 3:DISPLAY
4.EXIT 3
```

```
Content of List
20 30
1:INSERT 2:DELETE 3:DISPLAY 4.EXIT
```

3b. Application: Scientific applications involve many polynomials with varied degree. A polynomial is an expression consisting of variables and coefficients. Choose an appropriate data structure to represent the polynomial and perform addition of two polynomials.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int c;
        int p;
        struct node *link;
};
typedef struct node * NODE;

NODE getnode()
{
        NODE x;
        x = (NODE)malloc(sizeof(struct node));
        if (x == NULL)
        {
                printf ("Out of Memory");
                exit(0);
        }
        return x;
}
NODE insertr (int cf,int px,NODE first)
{
        NODE temp,pres;
        temp = getnode();
        temp->c = cf;
        temp->p = px;
        temp->link = NULL;
        if (first == NULL)
                return temp;
        pres = first;
        while (pres->link != NULL)
        {
                pres = pres->link;
        }
        pres->link = temp;
        return first;
}

NODE readpoly (NODE first)
{
        int i,n;
        int cf,px;
        printf ("Enter number of Terms\n");
        scanf ("%d",&n);
        for (i=1;i<=n;i++)
        {
                printf ("Enter Term %d\n",i);
                printf ("cf:");
                scanf ("%d",&cf);
                printf ("px:");
                scanf ("%d",&px);
                first = insertr (cf,px,first);
        }
        return first;
}

void display (NODE first)
{
        NODE temp;
        if (first == NULL)
        {
                printf ("Polynomial does not Exist\n");
                return;
        }
        temp = first;
        while (temp != NULL)
        {
                if(temp->c < 0)
                        printf ("%d",temp->c); // negative int
                else
                        printf ("+%d",temp->c); //positive int
                printf ("x^%d",temp->p); // x^exponent
                temp = temp->link;
        }
}

NODE addpoly (NODE poly1,NODE poly2,NODE poly)
{
        NODE pres_p1 = poly1,pres_p2 = poly2;
        while(pres_p1 != NULL && pres_p2 != NULL)
        {
                if (pres_p1->p > pres_p2->p)
                {
                        poly = insertr (pres_p1->c,pres_p1->p,poly);
                        pres_p1 = pres_p1->link;
                }
                else if (pres_p1->p < pres_p2->p)
                {
                        poly = insertr (pres_p2->c,pres_p2->p,poly);
                        pres_p2 = pres_p2->link;
                }
                else
                {
                        poly = insertr (pres_p1->c+pres_p2->c,pres_p1->p,poly);
```

```
            pres_p1 = pres_p1->link;
            pres_p2 = pres_p2->link;
        }
    }
    while (pres_p1 != NULL)
    {
        poly = insertr (pres_p1->c,pres_p1->p,poly);
        pres_p1 = pres_p1->link;
    }
    while (pres_p1 != NULL)
    {
        poly = insertr (pres_p2->c,pres_p2->p,poly);
        pres_p2 = pres_p2->link;
    }
    return poly;
}

int main()
{
    NODE
    poly1=NULL,poly2=NULL,poly=NULL; printf
    ("Enter Polynomial 1\n");
    poly1 = readpoly (poly1);
    printf ("\n");
    printf ("Enter Polynomial 2\n");
    poly2 = readpoly (poly2);
    printf ("\n");
    printf ("Polynomial 1 = ");
    display (poly1);
    printf ("\n");
    printf ("Polynomial 2 = ");
    display (poly2);
    printf ("\n");
    printf ("Result = ");
    poly = addpoly (poly1,poly2,poly);
    display (poly);
    printf ("\n");
    return 0;
}
```

OUTPUT

```
Enter Polynomial 1
Enter number of Terms
5
Enter Term 1
cf:4
px:1
Enter Term 2
cf:6
px:2
Enter Term 3
```

```
cf:3
px:3
Enter Term 4
cf:2
px:4
Enter Term 5
cf:1
px:5
Enter Polynomial 2
Enter number of Terms
5
Enter Term 1
cf:3
px:1
Enter Term 2
cf:6
px:2
Enter Term 3
cf:7
px:3
Enter Term 4
cf:5
px:4
Enter Term 5
cf:8
px:5
Polynomial                    1                   =
+4x^1+6x^2+3x^3+2x^4+1x^5  Polynomial  2
=   +3x^1+6x^2+7x^3+5x^4+8x^5   Result   =
+7x^1+12x^2+10x^3+7x^4+9x^5
```

4a. Program 4: Design, develop and execute a program in C to implement a doubly linked list where each node consists of integers. The program should support the following operations: a. Create a doubly linked list by adding each node at the front
b. Delete the node of a given data If it is found, otherwise display appropriate
message c. Display the contents of the list.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int info;
        struct node *llink;
        struct node *rlink;
};
typedef struct node *NODE;
NODE first = NULL,last = NULL;
void insert (int data)
{
        NODE newnode;
        newnode = (NODE)malloc(sizeof(struct node));
        newnode->info = data;
        newnode->llink = NULL;
        newnode->rlink = NULL;
        if(first == NULL)
        {
                first=last=newnode;
                return;
        }
        newnode->rlink = first;
        first->llink = newnode;
        first = newnode;
}
void delete (int key)
{
        int flag =0;
        NODE prev,cur,next;
        if (first == NULL)
        {
                printf ("List Empty\n");
                return;
        }
        if(first->rlink == NULL) // one node in the list
        {
                if (first->info == key)
                {
                        printf ("%d is deleted\n",first->info);
                        free (first);
                        first=last=NULL;
                        return ;
                }
        }
        if(key == first->info)
        {
                printf("\n%d is deleted\n",first->info);
                cur = first;
                first = first->rlink;
                first->llink = NULL;
                free(cur);
                cur=NULL;
                return;
        }
        if(key == last -> info)
        {
                printf("\n%d is deleted\n",last->info);
                cur = last;
                last = last->llink;
                last->rlink = NULL;
                free(cur);
                cur=NULL;
                return;
        }
        cur = first->rlink;
```

```
    while(cur!=last)
    {
        if(cur->info==key)
        {
            prev = cur->llink;
            next = cur->rlink;
            printf("\n%d is deleted\n",cur->info);
            prev->rlink = next;
            next->llink = prev;
            free(cur);
            cur = NULL;
            flag =1;
            break;
        }
        cur=cur->rlink;
    }
    if(flag==0)
        printf("\nKey not found\n");
}
void display ()
{
    NODE temp;
    if (first == NULL)
        printf ("List is Empty\n");
    else
    {
        printf ("Content of List\n");
        temp = first;
        while (temp != NULL)
        {
            printf ("%d\t",temp->info);
            temp = temp->rlink;
        }
        printf ("\n");
    }
}
int main ()
{
    int ch,data;
    for (;;)
    {
        printf ("1:INSERT 2:DELETE 3:DISPLAY 4:EXIT\n");
        scanf ("%d",&ch);
        switch (ch)
        {
            case 1: printf ("Enter the data\n");
                    scanf ("%d",&data);
                    insert (data);
                    break;
            case 2: printf ("Enter the data to be deleted\n");
                    scanf ("%d",&data);
```

```
                                        delete (data);
                                        break;
                        case 3: display ();
                                        break;
                        default: exit(0);
                }
        }
}
```

OUTPUT

```
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
1
Enter the data
10
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
1
Enter the data
20
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
1
Enter the data
30
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
1
Enter the data
40
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
1
Enter the data
50
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
3
Content of List
50 40 30 20 10
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
2
Enter the data to be deleted
50

50 is deleted
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
3
Content of List
40 30 20 10
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
2
Enter the data to be deleted
10

10 is deleted
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
```

```
3
Content of List
40 30 20
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
2
Enter the data to be deleted
30

30 is deleted
1:INSERT 2:DELETE 3:DISPLAY 4:EXIT
3
Content of List
40 20
```

4b. Application: Colleges maintain the student's records. As students enroll and graduate, the records have to be updated. It is necessary to access the student's data whenever required by college. Using suitable data structures develop a program for above scenario.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
        char name[30];
        char usn[20];
        int sem;
        struct node *left;
        struct node *right;
};
typedef struct node* NODE;
NODE insertFront (NODE first)
{
        NODE newnode;
        char name [30];
        char usn [20];
        int sem;
        newnode = (NODE)malloc(sizeof(struct node));
        newnode->left = newnode->right = NULL;
        printf ("Enter student details: \n");
        printf ("Name:\n");
        scanf ("%s", name);
        printf ("USN:\n");
        scanf ("%s",usn);
        printf ("Semester:\n");
        scanf ("%d",&sem);
        strcpy (newnode->name, name);
        strcpy (newnode->usn, usn);
        newnode->sem = sem;
        if (first == NULL)
        {
                first = newnode;
                return first;
        }
        else
        {
                newnode->right = first;
        }
        first->left=newnode;
        first = newnode;
        return first;
}
void search (NODE first)
{
        NODE pres;
        char usn [20];
        if (first == NULL)
        {
                printf ("Empty Database\n");
                return;
        }
        printf ("Enter USN to be searched: ");
        scanf ("%s", usn);
        pres = first;
        while (pres != NULL && strcmp (usn, pres->usn) != 0)
                pres = pres->right;
        if (pres == NULL)
                printf ("Student not found\n");
        else
                printf ("Student found\n");
}
void display (NODE first)
{
        if (first == NULL)
                printf ("Empty Database\n");
        else
        {
                printf ("Student details: \n");
                printf ("Name\t\tUSN\t\t\tSemester\n");
                while (first != NULL)
                {
                        printf ("%s\t\t%s\t\t%d\n", first->name, first->usn, first->sem);
                        first = first->right;
                }
        }
}
int main ()
{
        int ch;
        NODE first = NULL;
        for (;;)
        {
                printf ("1:INSERT 2:SEARCH 3:DISPLAY\n");
                scanf ("%d", &ch);
```

```
                    switch (ch)
                    {
                              case 1 : first = insertFront (first);
                                        break;
                              case 2 : search (first);
                                        break;
                              case 3 : display (first);
                                        break;
                              default : exit(0);
                    }
          }
          return 0;
}
```

OUTPUT

```
1:INSERT 2:SEARCH 3:DISPLAY
2
Empty Database
1:INSERT 2:SEARCH 3:DISPLAY
3
Empty Database
1:INSERT 2:SEARCH 3:DISPLAY
1
Enter student details:
Name:
ABC
USN:
1DS18CS710
Semester:
3
1:INSERT 2:SEARCH 3:DISPLAY
1
Enter student details:
Name:
XYZ
USN:
1DS16CS432
Semester:
6
1:INSERT 2:SEARCH 3:DISPLAY
1
Enter student details:
Name:
LMN
USN:
1DS15CS185
Semester:
2
1:INSERT 2:SEARCH 3:DISPLAY
1
Enter student details:
```

```
Name:
RST
USN:
1DS19CS742
Semester:
1
1:INSERT 2:SEARCH 3:DISPLAY
3
Student details:
Name USN Semester
RST 1DS19CS742 1
LMN 1DS15CS185 2
XYZ 1DS16CS432 6
ABC 1DS18CS710 3
1:INSERT 2:SEARCH 3:DISPLAY
2
Enter USN to be searched:
1DS18CS710 Student found
1:INSERT 2:SEARCH 3:DISPLAY
2
Enter USN to be searched:
1DS13CS432 Student not found
```

5a. Program 5: Design, develop and execute a program in C to create a Binary tree using arrays and display the tree.

```
#include <stdio.h>
#include <stdlib.h>
#define sz 100
void bt (int a[sz],int ele)
{
        int c,p,i;
        if (a[0] == '\0')
        {
                a[0] = ele;
                return;
        }
        c = 0;
        while (a[c] != '\0')
        {
                p = c;
                if (ele < a[c])
                        c = 2*c+1;
                else
                        c = 2*c+2;
        }
        if (ele < a[p])
                c = 2*p+1;
        else
                c = 2*p+2;
        a[c] = ele;
        printf ("Constructed Binary Tree is \n");
        for (i=0;i<sz;i++)
                if (a[i] != '\0')
                        printf ("a[%d]==>>%d\n",i,a[i]);
}
int main ()
{
        int n,a[sz],i,ele;
        for (i=0;i<sz;i++)
                a[i] = '\0';
        printf ("Enter the no of Data to Binary Tree\n");
        scanf ("%d",&n);
        printf ("Enter the Data to Binary Tree\n");
        for (i=0;i<n;i++)
        {
                scanf ("%d",&ele);
                bt (a,ele);
        }
}
```

OUTPUT

```
Enter the no of Data to Binary Tree
7
Enter the Data to Binary Tree
6 1 4 9 7 8 18
Constructed Binary Tree is
a[0]==>>6
a[1]==>>1
Constructed Binary Tree is
a[0]==>>6
a[1]==>>1
a[4]==>>4
Constructed Binary Tree is
a[0]==>>6
a[1]==>>1
a[2]==>>9
a[4]==>>4
Constructed Binary Tree is
a[0]==>>6
a[1]==>>1
a[2]==>>9
a[4]==>>4
a[5]==>>7
Constructed Binary Tree is
a[0]==>>6
a[1]==>>1
a[2]==>>9
a[4]==>>4
a[5]==>>7
a[12]==>>8
Constructed Binary Tree is
a[0]==>>6
```

```
a[1]==>>1
a[2]==>>9
a[4]==>>4
a[5]==>>7
a[6]==>>18
a[12]==>>8
```

5b. Application: Compilers use expression trees to represent mathematical expressions where the leaf nodes represent the operands and the internal nodes represent the operators. Develop a program to evaluate such an expression tree with non-negative integers as operands and the arithmetic operators ‘+’, ‘-‘, ‘*’and ‘/’.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int main ()
{
        char a[100],ch;
        int t1,t2,res,i,count=0;
        printf ("Enter Expression Tree of the form (a+b*(c/d)\n");
        scanf ("%s",a);
        for (i=0;i<strlen(a);i++)
        {
                ch = a[i];
                if (isdigit(ch))
                        break;
                else
                        count = count+1;
        }
        count = count-1;
        while (count >= 0)
        {
                t1 = a[2*count+1]-48;
                t2 = a[2*count+2]-48;
                if (a[count] == '/')
                        res = t1/t2;
                else if (a[count] == '*')
                        res = t1*t2;
                else if (a[count] == '+')
                        res = t1+t2;
                else if (a[count] == '-')
                        res = t1-t2;
                else
                        printf ("Invalid Expression\n");
                a[count] = res+48;
                count = count-1;
        }
        printf ("Evaluated Result from Expression Tree is %d\n",res);
}
```

OUTPUT

```
Enter Expression Tree
*+/4342
Evaluated Result from Expression Tree is 14
Program ended with exit code: 0
```

6a. Program 6: Design, develop and execute a program in C to create a Binary search tree and perform preorder traversals.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int info;
        struct node *left;
        struct node *right;
};
typedef struct node *NODE;
NODE insert (NODE root,int item)
{
        NODE newnode,prev = NULL,pres;
        newnode = (NODE)malloc(sizeof(struct node));
        newnode->info = item;
        newnode->left = newnode->right = NULL;
        if (root == NULL)
        {
                root = newnode;
                return root;
        }
        pres = root;
        while (pres != NULL)
        {
                prev = pres;
                if (item < pres->info)
                        pres = pres->left;
                else if (item > pres->info)
                        pres = pres->right;
                else
                {
                        printf ("Duplication of Item not allowed\n");
                        return root;
                }
        }
        if (item < prev->info)
                prev->left = newnode;
        else
                prev->right = newnode;
        return root;
}
void Preorder (NODE root)
{
        if (root != NULL)
        {
                printf("%d\t",root->info);
                Preorder (root->left);
                Preorder (root->right);
        }
}
int main ()
{
        NODE root = NULL;
        int ch,item;
        for (;;)
        {
                printf ("\n1:INSERT 2:PREORDER\n");
                scanf ("%d",&ch);
                switch (ch)
                {
                        case 1: printf ("Enter the Item\n");
                                scanf ("%d",&item);
                                root = insert(root,item);
                                break;
                        case 2: if (root == NULL)
                                        printf ("Empty Tree\n");
                                else
                                {
                                        printf ("Preorder:\n");
                                        Preorder(root);
                                }
                                break;
                        default: exit (0);
                }
        }
}
```

OUTPUT

```
1:INSERT 2:PREORDER
2
Empty Tree
1:INSERT 2:PREORDER
1
Enter the Item
5
1:INSERT 2:PREORDER
1
Enter the Item
70
1:INSERT 2:PREORDER
1
Enter the Item
23
1:INSERT 2:PREORDER
1
Enter the Item
```

```
70
Duplication of Item not allowed
1:INSERT 2:PREORDER
1
Enter the Item
65
1:INSERT 2:PREORDER
1
Enter the Item
54
1:INSERT 2:PREORDER
1
Enter the Item
18
1:INSERT
2:PREORDER 1
Enter the Item
25
1:INSERT
2:PREORDER 2
Preorder:
5 70 23 18 65 54 25
```

6b. Application : In a payroll management system, it is required to store the employee data (Employee ID, Employee name, Login Time) as one logs in to the system. At the end of the day, it is required to generate a report of all the employee who logged in that day in ascending order of the Employee ID. Develop a program to generate this report.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
        int eid;
        char ename [10];
        float lt;
        struct node *left;
        struct node *right;
};
typedef struct node *NODE;
NODE insert (NODE root,int eid,char ename [10],float lt)
{
        NODE newnode,prev,pres;
        newnode = (NODE)malloc(sizeof(struct node));
        newnode->left = newnode->right = NULL;
        newnode->eid = eid;
        strcpy (newnode->ename,ename);
        newnode->lt = lt;
        if (root == NULL)
        {
                root = newnode;
                return root;
        }
        pres = root;
        while (pres != NULL)
        {
                prev = pres;
                if (eid < pres->eid)
                        pres = pres->left;
                else if (eid > pres->eid)
                        pres = pres->right;
                else
                {
                        printf ("Duplicate\n");
                        return root;
                }
        }
        if (eid < prev->eid)
                prev->left = newnode;
        else
                prev->right = newnode;
        return root;
}
void inorder (NODE root)
{
        if (root != NULL)
        {
                inorder (root->left);
                printf
                ("%d\t%s\t\t%.2f\n",root->eid,root->ename,root->lt);
                inorder (root->right);
        }
}
int main ()
{
        NODE root = NULL;
        int ch,eid;
        char ename [10];
        float lt;
        for (;;)
        {
                printf ("1:INSERT 2:INORDER\n");
                scanf ("%d",&ch);
                switch (ch)
                {
                        case 1: printf ("Enter Employee Details:\n");
                                printf ("Employee ID\n");
                                scanf ("%d",&eid);
                                printf ("Employee Name\n");
                                scanf ("%s",ename);
                                printf ("Login Time\n");
                                scanf ("%f",&lt);
                                root = insert(root, eid, ename, lt);
```

```
                    break;
            case 2: if (root == NULL)
                            printf ("Employee Details Absent\n");
                    else
                    {
                            printf("Eid\tEname\tLT\n");
                            inorder(root);
                    }
                    break;
            default: exit(0);
        }
    }
}
```

OUTPUT

```
1:INSERT 2:INORDER
2
Employee Details Absent
1:INSERT 2:INORDER
1
Enter Employee Details:
Employee ID
111
Employee Name
ABC
Login Time
10.45
1:INSERT 2:INORDER
1
Enter Employee Details:
Employee ID
222
Employee Name
XYZ
Login Time
8.15
1:INSERT
2:INORDER 1
Enter Employee Details:
Employee ID
333
Employee Name
LMN
Login Time
2.30
1:INSERT
2:INORDER 2
Eid Ename LT
111 ABC 10.45
222 XYZ 8.15
333 LMN 2.30
```